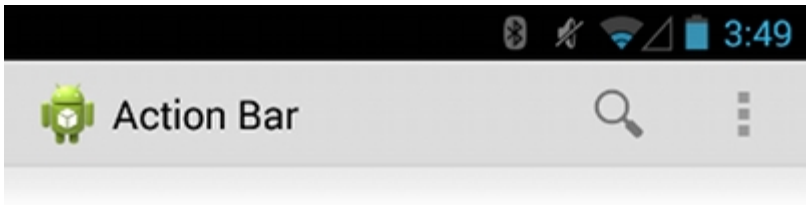
Android 官方开发教程中文版

添加操作栏

添加操作栏

操作栏是你能为你的 Activity 实现的重要的设计元素之一，它提供了几种用户界面特性，使你的应用能够保持和其它应用的一致性，以便用户很快熟悉它。主要功能包括：

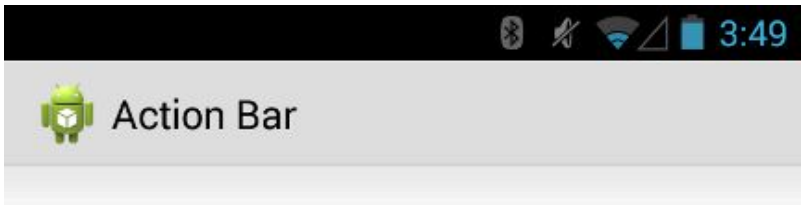
1. 用专门的空间为你的应用指定标识并且指示出用户在应用中的位置。
2. 以可预测的方式访问重要的操作（如搜索）
3. 支持导航和视图切换（使用选项卡或下拉列表）



本课程提供了关于操作栏基础知识的快速指南，要获得关于操作栏各种特性的更多信息，请参考“操作栏”指南。

设置操作栏

操作栏的最基本形式是为 Activity 显示标题以及在标题左边显示应用图标。即使是这种简单的形式，操作栏也有利于通知用户当前的位置，以及为你的应用保持一致性。



操作栏包括一个应用图标和 Activity 标题

设置一个基本的操作栏需要你的应用使用一个 Activity 主题并启用操作栏。如何获取这样一个主题取决于你的应用支持的最低 Android 版本。因此这节课根据你的应用支持的 Android 最低版本分为两个部分。

仅支持 Android3.0 及以上版本

从 Android3.0 开始（API 级别 11），操作栏被包含在所有使用 Theme.Holo（或它的派生类）主题的 Activity 中，当 targetSdkVersion 或 minSdkVersion 的值大于等于 11 时，Theme.Holo 是默认主题。

因此，要在 Activity 中添加操作栏，只需简单地把这两个属性之一的值设为 11 或更高就可以了，如：

```
<manifest ... >
    <uses-sdk android:minSdkVersion="11" ... />
```

```
    ...
</manifest>
```

注意：*如果你创建了自定义的主题，确保它的父类是 Theme.Holo 主题之一。详情请参考“**操作栏样式**”。*

现在你的应用使用了 Theme.Holo 主题，并且所有的 Activity 都会显示一个操作栏，就是这样。

支持 Android2.1 及以上版本

在比 Android3.0 更低的旧版本（低至 Android2.1）中添加操作栏，需要你的应用包含 Android 支持库。

现在就阅读“安装支持库”文档并安装 v7 appcompat 库（一旦你下载了库包，按照说明添加库资源）。

当你把支持库集成到项目中之后：

1. 更新你的 Activity 让它继承 ActionBarActivity，如：

   ```
   public class MainActivity extends ActionBarActivity { ... }
   ```

2. 在清单文件中，更新 <application> 元素或指定的 <activity> 元素，使用 Theme.AppCompat 之一做主题，如：

   ```
   <activity android:theme="@style/Theme.AppCompat.Light" ... >
   ```

注意：*如果你创建了自定义的主题，确保它的父类是 Theme.AppCompat 主题之一。详情请参考“**操作栏样式**”。*

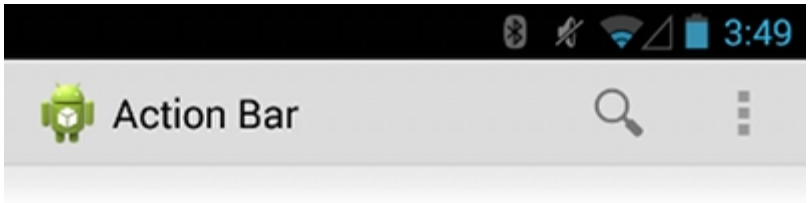
现在当运行在 Android2.1 或更高版本上时，你的 Activity 也包含操作栏了。

记住在清单文件中正确设置你支持的 API 级别。

```
<manifest ... >
    <uses-sdk android:minSdkVersion="7"   android:targetSdkVersion="18" />
    ...
</manifest>
```

添加操作按纽

操作栏允许你添加与应用当前上下文相关的重要操作项。它们作为操作按纽以图标或文本直接出现在操作栏上。操作栏上无法容纳或不太重要的按纽会隐藏在“更多操作”中。



一个包含了“搜索”按纽和“更多操作”（译注：三个点的图标）的操作栏

在 XML 中指定操作

所有操作按纽和“更多操作”中的其它项都在“菜单资源”的 XML 文件中定义。要添加操作按纽，在项目的 res/menu 目录下创建一个新的 XML 文件。

为每个要添加到操作栏上的项添加一个<item>元素，如：

res/menu/main_activity_actions.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
```

```
    <!-- Search, should appear as action button -->
    <item android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"
        android:showAsAction="ifRoom" />
    <!-- Settings, should always be in the overflow -->
    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:showAsAction="never" />
</menu>
```

这表示当操作栏的空间足够时，“搜索”操作会作为操作按钮出现在操作栏上，而“设置”操作则总是隐藏在“更多操作”中。（默认情况下，所有操作都会隐藏在“更多操作”中，但是为每个操作显式声明设计目标是一个好的习惯）。

icon 属性需要一个图片的资源 ID，@drawable 后面的名称必须是储存在 res/drawable 目录下的一个点阵图片名称。例如，"@drawable/ic_action_search"引用了 ic_action_search.png 文件。同样地，title 属性使用了定义在 res/values 目录下的 XML 文件中的字符串资源，这一点我们在“创建简单用户界面”中已经讨论过。

注意：*当为你的应用创建图标或其它点阵图时，重要的是要提供多个版本为不同的分辨率进行优化。我们将在“**支持不同屏幕**”中详细讨论。*

如果你的应用使用了支持库以兼容 Android2.1，在 android:命名空间中将不能使用 showAsAction 属性，你必须定义你自己的 XML 命名空间并使用这个命名空间作为属性前缀（一个自定义的 XML 命名空间应该基于你的应用名称，它也可以是任意一个名称，并且只能在你定义它的文件内访问）。例如：

res/menu/main_activity_actions.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:yourapp="http://schemas.android.com/apk/res-auto" >
    <!-- Search, should appear as action button -->
    <item android:id="@+id/action_search"
        android:icon="@drawable/ic_action_search"
        android:title="@string/action_search"
        yourapp:showAsAction="ifRoom"  />
    ...
</menu>
```

添加操作到操作栏上

要把菜单项放在操作栏上，实现 Activity 的 onCreateOptionsMenu()回调函数，把菜单资源加载到给定的菜单对象中，如：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu items for use in the action bar
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main_activity_actions, menu);
    return super.onCreateOptionsMenu(menu);
```

```
}
```

响应操作按钮

当用户按下操作按钮或“更多操作”中的其它项，系统会调用 Activity 的 onOptionsItemSelected()回调方法。在实现这个方法时，对给定的 MenuItem 调用 getItemId()方法以确定哪个菜单项被按下。getItemId()返回的 ID 对应你在<item>元素 android:id 属性中定义的值。

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle presses on the action bar items
    switch (item.getItemId()) {
        case R.id.action_search:
            //openSearch();
            return true;
        case R.id.action_settings:
            //openSettings();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

译注：*原文中没有给出 openSearch()和 openSettings()两个方法的代码，所以这里把它们注释了。*

为低级 Activity 添加 Up 按纽

在你的应用中，所有屏幕都不是应用的主入口（这些 Activity 并没有“home”屏幕）。应该给用户提供一种方法，通过按下操作栏上的 Up 按纽可以导航到应用逻辑层次中“父屏幕”。

当运行在 Android4.1（API 级别 16）或更高版本上，或是从支持库中使用 ActionBarActivity 时，要执行“向上”导航，只需简单地在清单文件中声明父 Activity 并在操作栏中启用“向上”按纽。

以下是如何在清单文件中声明 Activity 的父项的例子：

```
<application ... >
    ...
    <!-- The main/home activity (it has no parent activity) -->
    <activity
        android:name="com.example.myfirstapp.MainActivity" ...>
        ...
    </activity>
    <!-- A child of the main activity -->
    <activity
        android:name="com.example.myfirstapp.DisplayMessageActivity"
        android:label="@string/title_activity_display_message"
        android:parentActivityName="com.example.myfirstapp.MainActivity" >
```

```
        <!-- Parent activity meta-data to support 4.0 and lower -->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.myfirstapp.MainActivity" />
    </activity>
</application>
```

为“向上”按钮启用应用图标，要调用 setDisplayHomeAsUpEnabled()方法：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_displaymessage);

    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    // If your minSdkVersion is 11 or higher, instead use:
    // getActionBar().setDisplayHomeAsUpEnabled(true);
}
```

系统现在知道 MainActivity 是 DisplayMessageActivity 的父 Activity，当用户按下“向上”按钮时，系统将会导航到父 Activity，你不需要处理“向上”按钮的按下事件。

要获取有关导航的更多信息，请参考“**提供向上导航**”。

操作栏样式

操作栏为你的用户提供了熟悉和可预见的方式执行操作和导航应用，但这并不意味着它看起来要和其它应用一模一样。如果你想要操作栏的风格能更好地适合你的产品品牌，利用 Android 的“样式和主题”资源很容易做到这一点。

Android 提供了几个内置的 Activity 主题，包括“Dark”和“Light”操作栏样式。你还可能扩展这些主题以进一步的定义操作栏的外观。

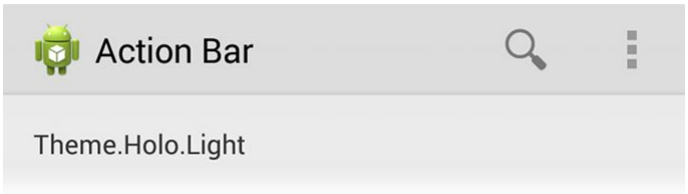
注意：如果你正在为操作栏使用支持库 API，你必须使用（或重写）Theme.AppCompat 系列样式（而不是只能用于 API 级别 11 或更高版本的 Theme.Holo 系列样式）。要做到这点，你声明的每个样式属性都必须定义两次：一次用于平台的样式属性（android:属性），一次用于包含在支持库中的样式属性（appcompat.R.attr 属性——这些属性实际上是你的应用的上下文）。详情见下面的示例。

使用 Android 主题

Android 包括两个决定操作栏颜色的 Activity 基本主题：

- “Theme.Holo”用于“Dark”主题
- “Theme.Holo.Light”用于“Light”主题



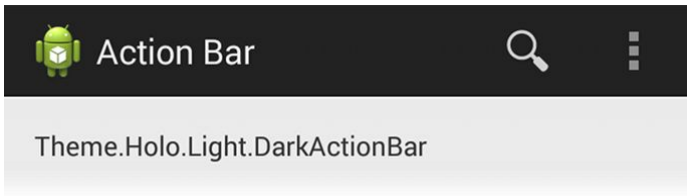


通过在清单文件中指定<application>元素或某个<activity>元素的 android:theme 属性，你可以把这些主题应用到你整个 App 或是个别的 Activity 上。

例：

```
<application android:theme="@android:style/Theme.Holo.Light" ... />
```

你还可以使用“Theme.Holo.Light.DarkActionBar”主题在亮色的 Activity 上显示暗色的操作栏。



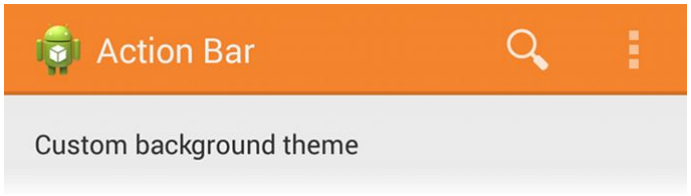
当使用支持库时，你必须用“Theme.AppCompat”系列的主题替换上面的主题。

- “Theme.AppCompat”用于“Dark”主题
- “Theme. AppCompat.Light”用于“Light”主题
- “Theme.AppCompat.Light.DarkActionBar”用于带暗色操作栏的亮色主题。

确保你的操作栏图标和操作栏的颜色有合适地反差。“操作栏图标包”中包括了用于“Holo dark”和“Holo light”两种风格操作栏的标准图标。

自定义背景

要改变操作栏的背景，为你的 Activity 创建一个自定义的主题并重写 actionBarStyle 属性。这个属性指向另一个样式，在这个样式中，用 background 属性为操作栏背景指定一个可绘制资源。



如果你的应用使用了 navigation tabs 或 split action bar，你也可以分别使用 backgroundStacked 和 backgroundSplit 属性指定它们的背景。

警告：*为你的主题和样式声明一个适当的父主题以继承它们的风格，这很重要。否则，你的操作栏将会缺少很多样式属性，除非你自己显式地定义它们。*

仅用于 Android3.0 或更高版本

当仅支持 Android3.0 或更高版本时，你可以像这样定义操作栏的背景：

res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
```

```
            parent="@android:style/Theme.Holo.Light.DarkActionBar">
        <item name="android:actionBarStyle">@style/MyActionBar</item>
    </style>

    <!-- ActionBar styles -->
    <style name="MyActionBar"
            parent="@android:style/Widget.Holo.Light.ActionBar.Solid.Inverse">
        <item name="android:background">@drawable/actionbar_background</item>
    </style>
</resources>
```

译注：上面 XML 文件中加粗字体部份原文是 parent="@style/..."，这表示引用自定义样式，是错误的，应改成@android:style/，表示引用系统样式。

要应用到整个应用或某个 Activity：

```
<application android:theme="@style/CustomActionBarTheme" ... />
```

用于 Android2.1 或更高版本

当使用支持库时，要应用和上面同样的主题必须如下所示：

res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
            parent="@android:style/Theme.AppCompat.Light.DarkActionBar">
        <item name="android:actionBarStyle">@style/MyActionBar</item>

        <!-- Support library compatibility -->
        <item name="actionBarStyle">@style/MyActionBar</item>
    </style>

    <!-- ActionBar styles -->
    <style name="MyActionBar"
            parent="@android:style/Widget.AppCompat.Light.ActionBar.Solid.Inverse">
                                                                    <item
name="android:background">@drawable/actionbar_background</item>

        <!-- Support library compatibility -->
        <item name="background">@drawable/actionbar_background</item>
    </style>
</resources>
```

译注：上面 XML 文件中加粗字体部份原文是 parent="@style/..."，这表示引用自定义样式，是错误的，应改成@android:style/，表示引用系统样式。

要应用到整个应用或某个 Activity：

```
<application android:theme="@style/CustomActionBarTheme" ... />
```

自定义文本颜色

修改操作栏的文本颜色，需要为每个文本元素单独重写属性。

- 操作栏标题文本：在自定义的 actionBarStyle 中，为 titeTextStyle 属性指定一个定义了 textColor 属性的样式。
 注意： *应用到 titleTextStyle 属性的自定义样式使用 TextAppearance.Holo.Widget.ActionBar.Title 做为父样式。*
- 操作栏选项卡：在 Activity 主题中重写 actionBarTabTextStyle。
- 操作按钮：在 Activity 主题中重写 actionMenuTextStyle。
 译注： *在 Eclipse 中添加颜色定义，在项目的 res/values 目录上单击右键，**New>Android XML File**，在文件中输入 colors，点击 **Finish**。为每个颜色定义添加一条<color name="颜色名称">#颜色代码</color>，颜色代码是 4 字节的 16 进制数，每个字节分别代表透明度、红、绿、蓝。*

仅用于 Android3.0 或更高版本

当仅支持 Android3.0 或更高版本时，你的样式 XML 文件类似于：

res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
            parent="@style/Theme.Holo">
        <item name="android:actionBarStyle">@style/MyActionBar</item>
                                                                    <item
name="android:actionBarTabTextStyle">@style/MyActionBarTabText</item>
        <item name="android:actionMenuTextColor">@color/actionbar_text</item>
    </style>

    <!-- ActionBar styles -->
    <style name="MyActionBar"
            parent="@style/Widget.Holo.ActionBar">
        <item name="android:titleTextStyle">@style/MyActionBarTitleText</item>
    </style>

    <!-- ActionBar title text -->
    <style name="MyActionBarTitleText"
            parent="@style/TextAppearance.Holo.Widget.ActionBar.Title">
        <item name="android:textColor">@color/actionbar_text</item>
    </style>

    <!-- ActionBar tabs text styles -->
    <style name="MyActionBarTabText"
            parent="@style/Widget.Holo.ActionBar.TabText">
        <item name="android:textColor">@color/actionbar_text</item>
    </style>
</resources>
```

译注： *以上所有的@style 请改为@android:style。*

用于 Android2.1 或更高版本

当使用支持库时，你的样式 XML 文件类似于：

res/values/themes.xml

```
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
            parent="@style/Theme.AppCompat">
        <item name="android:actionBarStyle">@style/MyActionBar</item>
                                                                    <item
name="android:actionBarTabTextStyle">@style/MyActionBarTabText</item>
        <item name="android:actionMenuTextColor">@color/actionbar_text</item>

        <!-- Support library compatibility -->
        <item name="actionBarStyle">@style/MyActionBar</item>
        <item name="actionBarTabTextStyle">@style/MyActionBarTabText</item>
        <item name="actionMenuTextColor">@color/actionbar_text</item>
    </style>

    <!-- ActionBar styles -->
    <style name="MyActionBar"
            parent="@style/Widget.AppCompat.ActionBar">
        <item name="android:titleTextStyle">@style/MyActionBarTitleText</item>

        <!-- Support library compatibility -->
        <item name="titleTextStyle">@style/MyActionBarTitleText</item>
    </style>

    <!-- ActionBar title text -->
    <style name="MyActionBarTitleText"
            parent="@style/TextAppearance.AppCompat.Widget.ActionBar.Title">
        <item name="android:textColor">@color/actionbar_text</item>
        <!-- The textColor property is backward compatible with the Support Library -->
    </style>

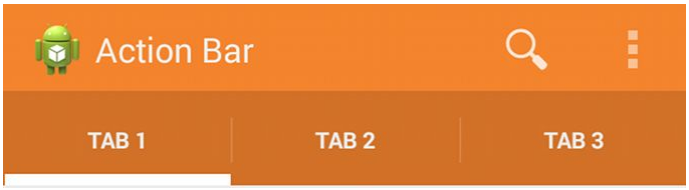
    <!-- ActionBar tabs text -->
    <style name="MyActionBarTabText"
            parent="@style/Widget.AppCompat.ActionBar.TabText">
        <item name="android:textColor">@color/actionbar_text</item>
        <!-- The textColor property is backward compatible with the Support Library -->
    </style>
</resources>
```

译注： *以上所有的@style 请改为@android:style。*

自定义选项卡指示器

译注：*本节及后面两节的内容涉及到如何添加导航选项卡，Android 开发新手可以先跳过。*

要改变导航栏选项卡的指示器，为你的 Activity 创建一个自定义的主题并重写 actionBarTabStyle 属性。这个属性指向另一个样式，在这个样式中，为 background 属性指定一个状态列表可绘制资源。



注意：状态列表可绘制资源对显示处于选中状态的选项卡和其它选项卡不同的背景非常重要。要获得更多如何创建可绘制资源以处理按纽多种状态的信息，请参考 State List 文档。

这里是用状态列表可绘制资源为操作栏选项卡的几种状态定义不同背景图片的例子。

res/drawable/actionbar_tab_indicator.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

<!-- STATES WHEN BUTTON IS NOT PRESSED -->

    <!-- Non focused states -->
    <item android:state_focused="false" android:state_selected="false"
        android:state_pressed="false"
        android:drawable="@drawable/tab_unselected" />
    <item android:state_focused="false" android:state_selected="true"
        android:state_pressed="false"
        android:drawable="@drawable/tab_selected" />

    <!-- Focused states (such as when focused with a d-pad or mouse hover) -->
    <item android:state_focused="true" android:state_selected="false"
        android:state_pressed="false"
        android:drawable="@drawable/tab_unselected_focused" />
    <item android:state_focused="true" android:state_selected="true"
        android:state_pressed="false"
        android:drawable="@drawable/tab_selected_focused" />


<!-- STATES WHEN BUTTON IS PRESSED -->

    <!-- Non focused states -->
    <item android:state_focused="false" android:state_selected="false"
        android:state_pressed="true"
```

```
            android:drawable="@drawable/tab_unselected_pressed" />
        <item android:state_focused="false" android:state_selected="true"
            android:state_pressed="true"
            android:drawable="@drawable/tab_selected_pressed" />

        <!-- Focused states (such as when focused with a d-pad or mouse hover) -->
        <item android:state_focused="true" android:state_selected="false"
            android:state_pressed="true"
            android:drawable="@drawable/tab_unselected_pressed" />
        <item android:state_focused="true" android:state_selected="true"
            android:state_pressed="true"
            android:drawable="@drawable/tab_selected_pressed" />
    </selector>
```

仅用于 **Android3.0** 或更高版本

当仅支持 Android3.0 或更高版本时，你的样式 XML 文件类似于：

res/values/themes.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <resources>
        <!-- the theme applied to the application or activity -->
        <style name="CustomActionBarTheme"
                parent="@style/Theme.Holo">
            <item name="android:actionBarTabStyle">@style/MyActionBarTabs</item>
        </style>

        <!-- ActionBar tabs styles -->
        <style name="MyActionBarTabs"
                parent="@style/Widget.Holo.ActionBar.TabView">
            <!-- tab indicator -->
                                                                    <item
    name="android:background">@drawable/actionbar_tab_indicator</item>
        </style>
    </resources>
```

用于 **Android2.1** 或更高版本

当使用支持库时，你的样式 XML 文件类似于：

res/values/themes.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <resources>
        <!-- the theme applied to the application or activity -->
        <style name="CustomActionBarTheme"
                parent="@style/Theme.AppCompat">
            <item name="android:actionBarTabStyle">@style/MyActionBarTabs</item>
```

```
        <!-- Support library compatibility -->
        <item name="actionBarTabStyle">@style/MyActionBarTabs</item>
    </style>

    <!-- ActionBar tabs styles -->
    <style name="MyActionBarTabs"
           parent="@style/Widget.AppCompat.ActionBar.TabView">
        <!-- tab indicator -->
                                                                    <item
name="android:background">@drawable/actionbar_tab_indicator</item>

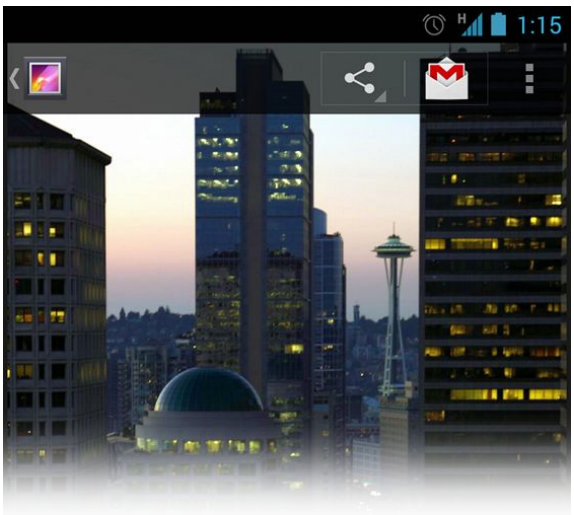
        <!-- Support library compatibility -->
        <item name="background">@drawable/actionbar_tab_indicator</item>
    </style>
</resources>
```

操作栏覆盖模式

默认情况下，操作栏位于 Activity 的顶部，这会稍微减少 Activity 的布局空间。在和用户交互的过程中，如果你想隐藏或显示操作栏，你可以调用 ActionBar 的 hide()和 show()方法，但这样会导致 Activity 按新的尺寸重新计算及绘制布局。

要避免当隐藏或显示操作栏时重新调整布局大小，你可以启用操作栏的覆盖模式。在覆盖模式中，Activity 布局将使用没有操作栏时的所有空间，并且系统将把操作栏绘制在“前方”，这会遮挡住布局顶部的部分区域，但当操作栏隐藏或显示时，系统不需要重新调整布局大小并可以无缝切换。

提示：如果你希望操作栏后面的部分可见，你可以创建一个背景部分透明的操作栏自定义样式，如下图所示。如何定义操作栏背景，见“**操作栏样式**”



启用覆盖模式

要启用操作栏的覆盖模式，你需要声明一个自定义主题以继承现有的操作栏主题，并把 android:windowActionBarOverlay 属性设为 true。

仅用于 Android3.0 或更高版本

如果你 minSdkVersion 设定为 11 或更高，你可以用 Theme.Holo（或它的派生类）做为父主题来自定义一个主题，例如：

```
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
            parent="@android:style/Theme.Holo">
        <item name="android:windowActionBarOverlay">true</item>
    </style>
</resources>
```

用于 Android2.1 或更高版本

如果你的应用使用了支持库以兼容低于 Android3.0 的版本，你应该用 Theme.AppCompat（或它的派生类）做为父主题来自定义一个主题，例如：

```
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
            parent="@android:style/Theme.AppCompat">
        <item name="android:windowActionBarOverlay">true</item>

        <!-- Support library compatibility -->
        <item name="windowActionBarOverlay">true</item>
    </style>
</resources>
```

要注意这个主题包括两个“windowActionBarOverlay”的定义，一个有 android 前缀，一个没有。有前缀的是用于平台中已经包含样式的 Android 版本，没有前缀的是用于从支持库中读取样式的旧 Android 版本。

指定布局的顶部边距

当操作栏处于覆盖模式时，布局中应该可见的部分可能会被遮挡，要确保这些内容任何时候都在操作栏下面，用 actionBarSize 指定的高度为视图的顶部增加外边距或内边距。如：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="?android:attr/actionBarSize">
    ...
</RelativeLayout>
```

如果你为操作栏使用了支持库，那么删除 android 前缀，如：

```
<!-- Support library compatibility -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="?attr/actionBarSize">
    ...
</RelativeLayout>
```

在本例中，?attr/actionBarSize 没有前缀，可以工作在所有版本下，包括 Android3.0 或更高版本。